Amendments to the Specification:

Please amend the specification in paragraphs [0005], [0045], and [0047] as follows:

[0005] A method and system for developing and executing applications at an abstract design level is disclosed. In one embodiment of the present invention, a visual modeling or assembly tool is used to capture application logic, independently from the underlying technologies and hardware and software infrastructure, and to deploy it onto an execution platform dynamically. The execution platform is responsible for providing access to a variety of types of external client devices to execute the application logic and receive responses in formats suitable for the external client devices.

[0045] In a web-based application, the typical way of retrieving information from the user is on a page-by-page basis. This tends to significantly complicate the implementation of the software application by breaking down logical flows into disjoint chunks of logic spread out and embedded in several web pages. In addition, session state management is both manual and difficult. However, with the pause/resume mechanism, one can insert read functions as steps in a well coherent and logical manner by which to retrieve additional information from the user. Once the execution engine encounters each read, it will pause execution, serve the object to be read/updated to the user, and waits until the user completes the task and resume processing. On resumption, the execution engine rebuilds the execution stack to where it left of, and continues execution of the subsequent steps. The execution engine also maintains intermediate stack states to support the Back button available on most browsers.

[0047] In summary, the VE system provides a framework for separating the interface code from the application logic to ease the development of a business software. This facilitates the customization of the interface code to support current and future types of external entities, independent from the customized development of the application logic. The VE system also provides visual modeling tools that comply with industry standard

notation to specialize the application logic side of the framework into business applications, while completely eliminating the low-level coding and required technology skill sets, and allowing the end-user to directly interact[s] with and manipulate[s] modeled objects. The VE system blurs the line between development time and runtime, thus providing the end-user with customization and personalization tools, and hence, shifting some of the development activities from the application developer to the end-user.

Please amend the specification in paragraphs [0018], [0036], and [0040] as follows:

[0018] By examining computer programs, and especially business applications, it is discovered that all applications are made of two types of code: code that captures the application logic, and code that interfaces the application logic with external entities. A sample ~~Java~~ program in the JAVA programming language shown below is a simple implementation of a predetermined application logic to print the result of 2 plus 2 to the command shell,

[0036] FIG. 8 graphically illustrates an execution platform 300 for the VE system. The execution platform is responsible for hosting the application logic (e.g., formulas 340, object definitions 350, and high-level constructs 360 such as processes, rules, spreadsheets, neural networks). Further, the execution platform 300 is responsible for providing access to external client devices 380 to submit an external request for executing application logic and receiving a response in a format appropriate to the external client device 380. The external client device can be a web browser, a WAP phone, a PDA, or a device implemented with voice XML technologies. It is also contemplated that another system using web servers can also be communicating with the execution platform. Listening, receiving, and responding to external requests are the responsibilities of the network server 310 (e.g. HTTP, RPC, OMG CORBA integration standard, or MICROSOFT DCOM communication server) subsystem. A storage device management subsystem 370 of the execution platform 300 may also interact with

different storage device schemas in one or more storage devices 390 such as a database, an LDAP, or a file system. The storage device schemas are databases, columns, rows, or other entities used for storing information in a storage device, and they are created after the application logic is captured and for implementing the application logic.

[0040] Upon completing the execution of the application logic, the execution engine 330 will respond to the network server 310 with an execution response object. The network server 310 will use a client device management subsystem 320 for converting the response object to the appropriate format (E). Taking into consideration the external client device 380 type (e.g. ~~Internet Explorer, Netscape,~~ INTERNET EXPLORER web browser, NETSCAPE web browser, WAP phone, SOAP client), version, accepted mime types, and locale information, the client device management subsystem 320 is responsible for maintaining format information ("skins") for each object definition 350, deciding which content handler to use to convert a given object into an appropriate skin, and using a content handler to convert an object to a format suitable for the external client device 380. The network server 310 will then respond to the external client device request with the appropriately formatted response (F).

Please amend the specification in paragraphs [0034] follows:

[0034] FIG. 7 illustrates a process editor 200 for capturing a high-level structure (e.g., a process). Again, the UML notation is used to capture a process in an activity diagram. A process has a beginning 210, and an end 220, and at least one thread 230 of execution (or transition). Between the beginning 210 and end ~~220~~ (not shown), one or more activities 240, decisions 250, forks ~~260~~, states ~~270~~, and loops 280 govern the sequence of the execution of the process, thus indirectly control the desired application logic. Each activity specifies an appropriate formula 290 to be executed. In addition, decisions, loops, and states also use formulas to specify desired execution behaviors. Once the application logic is captured, it will be saved directly to an execution platform. In addition, the visual modeling tool provides a mechanism to deploy the captured application logic onto an execution platform. The process of deployment includes persisting the application logic

on the execution platform, and generating the appropriate storage system schemas to house persistent objects. The format of the generated schemas depends on the type of the storage device (e.g. relational schema and object-to-relational mapping for relational databases).